Angular Notes

1. What is Angular?

Angular is a TypeScript-based front-end framework developed by Google. It helps build Single Page Applications (SPAs) — web apps that load one HTML page and dynamically update content.

Key Features:
- Component-based architecture
- Two-way data binding
- Dependency Injection (DI)
- Directives and Pipes
- Routing and navigation
- Reactive programming with RxJS

2. Angular Architecture Overview

1. Modules: Collection of related components, directives, and services. Root module = AppModule.
2. Components: The basic building blocks of UI. Each component has HTML, CSS, and TS file.
3. Templates: Define the HTML structure for components.
4. Metadata: Decorators like @Component, @NgModule tell Angular how to process a class.
5. Data Binding: Connects TypeScript code and HTML view.
6. Directives: Used to modify DOM structure or behavior.
7. Services & Dependency Injection: Used for business logic and reusable code.
8. Routing: Used for navigation between components.

3. Components

Each component has HTML Template, TypeScript Class, CSS/SCSS, and a Decorator (@Component).

Example:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'My First Angular App';
}
```

4. Data Binding Types

1. Interpolation: {{ variable }} — Displays dynamic data in HTML.
2. Property Binding: [property]="value" — Binds property of an element.
3. Event Binding: (event)="method()" — Handles DOM events.
4. Two-way Binding: [(ngModel)]="variable" — Syncs data between component and template.

Example:

```
<input [(ngModel)]="name" placeholder="Enter name">
<p>Hello {{ name }}!</p>
```

5. Directives

- Structural Directives (*ngIf, *ngFor, *ngSwitch) — Change the DOM structure.
- Attribute Directives ([ngClass], [ngStyle]) — Change the appearance or behavior of elements.

- Example:

```
<p *ngIf="isLoggedIn">Welcome!</p>
<li *ngFor="let user of users">{{ user }}</li>
```

6. Services & Dependency Injection (DI)

A service is a class used to share data or logic across components.

Example:

```
import { Injectable } from '@angular/core';

@Injectable({ providedIn: 'root' })
export class DataService {
  getData() {
        return ['Angular', 'React', 'Vue'];
  }
}
```

Use in component:

```
constructor(private dataService: DataService) {}
ngOnInit() {
  this.courses = this.dataService.getData();
}
```

7. Routing

Used for navigation between pages.

Example:

```
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';

const routes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'about', component: AboutComponent },
  { path: '', redirectTo: '/home', pathMatch: 'full' }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule {}
```

8. Pipes

Used to transform data in templates.

Example:

```
<p>{{ title | uppercase }}</p>
<p>{{ date | date:'short' }}</p>

Custom Pipe Example:
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({ name: 'reverse' })
export class ReversePipe implements PipeTransform {
  transform(value: string): string {
        return value.split('').reverse().join('');
  }
}
```

9. Lifecycle Hooks

- ngOnInit() — Runs once after component initialization.
- ngOnChanges() — Runs when input properties change.
- ngDoCheck() — Custom change detection.
- ngAfterViewInit() — After component view is initialized.
- ngOnDestroy() — Cleanup before component is destroyed.

10. Angular CLI Commands

- ng new app-name — Create a new Angular app.
- ng serve — Run app in development mode.
- ng generate component comp-name — Create a new component.
- ng generate service service-name — Create a new service.
- ng build — Build the app for production.
- ng test — Run tests.

11. Reactive Forms

Reactive forms are model-driven forms.

Example:

```
import { FormGroup, FormControl } from '@angular/forms';

form = new FormGroup({
  name: new FormControl(''),
  email: new FormControl('')
});
```

HTML:

```
<form [formGroup]="form">
  <input formControlName="name">
  <input formControlName="email">
</form>
```

12. HTTP Client (API Calls)

Import HttpClientModule in app.module.ts.

Example:

```
import { HttpClient } from '@angular/common/http';

constructor(private http: HttpClient) {}

getUsers() {
  this.http.get('https://jsonplaceholder.typicode.com/users')
        .subscribe(data => console.log(data));
}
```

13. Observables & RxJS

Observable: Stream of data over time.
Used with HTTP, Forms, and Events.
RxJS provides operators like map(), filter(), subscribe().

Example:

```
this.http.get(url).subscribe(response => {
  console.log(response);
});
```

14. Difference Between AngularJS and Angular

AngularJS (JS-based) vs Angular (TypeScript-based):
- AngularJS uses MVC, Angular uses Component-based architecture.
- AngularJS supports only two-way binding, Angular supports both.
- Angular is mobile-friendly and modern, AngularJS is older.